University of Ljubljana
Faculty of computer and information science

Gjorgievska Sirma

Simulation and 3D visualization of physical phenomena on mobile devices

bachelor's thesis

undergraduate university study program
computer and information science

Advisor: prof. dr. Saša Divjak

Ljubljana 2014

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Gjorgievska Sirma

Simulacija in 3D vizualizacija fizikalnih pojavov na mobilnih napravah

diplomsko delo

univerzitetni program prve stopnje
računalništvo in informatika

Mentor: prof. dr. Saša Divjak

Ljubljana 2014

The text is formatted with the editor LaTeX.

[1] In agreement with the advisor, the candidate can issue his thesis together with the source code under some of the alternative licenses, which offer some of his rights to all: eg. Creative Commons, GNU GPL. In this case, here you should insert description of the license.

Faculty of computer and information science issues the following thesis:

Examine tools for computer simulation and 3D visualization of physical phenomena, in particular the tools xyZET, JavaXYZ and jsXYZ that provide such simulation in environments Motif, Java and JavaScript. Use jQuery and three.js framework and develop and test your application on mobile devices. Introduce the possibility of multi-user access to such simulations.

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

[2]V dogovorju z mentorjem lahko kandidat diplomsko delo s pripadajočo izvorno kodo izda tudi pod katero izmed alternativnih licenc, ki ponuja določen del pravic vsem: npr. Creative Commons, GNU GPL. V tem primeru na to mesto vstavite opis licence.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Proučite orodja za računalniško simulacijo in 3D vizualizacijo fizikalnih pojavov in se pri tem naslonite na orodja xyZET, Java-XYZ in jsXYZ, ki nudijo tako simulacijo v okoljih Motif, Javi in JavaScript. Uporabite ogrodja jQuery in three.js in razvijte ter preskusite aplikacijo na mobilnih napravah. Uvedite tudi možnost večuporabniškega dostopa do takih simulacij.

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Gjorgievska Sirma, z vpisno številko **63110392**, sem avtor diplomskega dela z naslovom: Simulacija in 3D vizualizacija fizikalnih pojavov na mobilnih napravah

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom prof. dr. Saša Divjak,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 8. septembra 2014 Podpis avtorja:

First of all, I would like to thank my mentor, prof. dr. Saša Divjak for his support and guidance. Despite all of his duties, he always found time for discussions and responded immediately to all of my questions and problems.

Moreover, I would like to thank Tilen and all of the other colleagues and friends who helped me during my studies.

I am especially grateful to my best friend Ana, for being with me through my hardest times. It was much easier to have discipline, to study all night and to have faith to myself when I have a friend like you beside me.

Furthermore, I want to express my gratitude to Robert, who encouraged me and expressed confidence in my abilities when I could only do the opposite. Thank you for all of your help and guiding my thesis in the right direction.

Finally, I have to thank my parents for their endless love and support throughout my life. Thank you for believing in me, even when it was hard for me to believe in myself. Thank you both for giving me the strength never to give up and always to chase my dreams. All I am today I owe to you.

This thesis is dedicated to my parents. For their endless love, support and encouragement.

Contents

Abstract

Povzetek

1 Introduction **1**

2 Overview of JavaXYZ **3**

2.1 Background 3

2.2 Physical model 6

2.2.1 Elastic forces for modeling spring action 7

2.2.2 Damping forces 7

2.2.3 Friction forces 8

2.2.4 Gravitation forces (external and reciprocal) 8

2.2.5 Coulomb forces (external and reciprocal) 8

2.2.6 Magnetic forces 9

2.3 Base algorithm 9

2.3.1 Sequence of commands in the step of calculating 9

2.4 Mathematical basics 10

3 Simulation tool jsXYZ **13**

3.1 JavaScript 14

3.2 Three.js 16

4 jQuery Mobile **19**

4.1 CSS Framework 20

4.2 Events 20
4.3 Icons 20
4.4 Methods 21
4.5 Reference 21
4.6 Widgets 23
4.6.1 Button Widget 23
4.6.2 Checkboxradio Widget 23
4.6.3 Radio buttons 25
4.6.4 Collapsibleset Widget 25
4.6.5 Controlgroup Widget 27
4.6.6 Dialog Widget 28
4.6.7 Listview Widget 29
4.6.8 Navbar Widget 30
4.6.9 Page Widget 31
4.6.10 Popup Widget 31
4.6.11 Slider Widget 33
4.6.12 Toolbar Widget 33

5 xyzMobile - Adjustment of jsXYZ for mobile devices 37
5.1 Support for mobile devices and tablets 38
5.2 Support for multiuser physical simulations on mobile devices . 49
5.2.1 WebSocket API 50
5.3 JavaScript commands in xyzMobile application 53

6 Conclusion 55

List of abbreviations

Abbreviation	Meaning
UI	User Interface
CSS	Cascading Style Sheets
HTML	Hyper Text Markup Language
CGI	Common Gateway Interface
API	Application Programming Interface
WebGL	Web Graphics Library
SVG	Scalable Vector Graphics
GPU	Graphics Processing Unite
DOM	Document Object Model
PNG	Portable Network Graphics
SD	Standard Definition
HD	High Definition
URL	Uniform Resource Locator

Abstract

Educational simulations are breaking out from traditional areas of use and emerging as an increasingly important tool for education and training. Even though, they are very practical tool for assisting in the traditional classrooms, they have so far been supported mostly on desktop environments and did not fulfill the desired widespread adoption. On the other hand, miniaturization and wireless communication advances have given us an ever- increasing array of portable devices, each with unique capabilities and form-factors, which students use to facilitate learning and enhance productivity in school. For this reason we have created the mobile application xyzMobile that is a tool for presenting physics simulations. It is supported by different devices, including mobile phones and tablets, taking advantage of the different display size, portability, and computational power characteristics of the devices. To make this application reachable not just from desktop environments, we used jQuery Mobile, the client side framework that is optimized for touch devices. We have also provided the multiuser support of the simulations in real-time, offering users the possibility to simultaneously manipulate shared real-time simulations. For that purpose we have implemented WebSockets, technology that is providing an interactive communication session between the server and the browser, facilitating live content and the creation of real-time simulations.

Keywords: simulations, mobile application, jQuery Mobile, WebSockets.

Povzetek

Izobraževalne simulacije se vedno bolj širijo iz tradicionalnih področij in postajajo pomembno orodje za izobraževanje. Čeprav so to zelo praktična orodja za pomoč v klasičnem izobraževanju, so bila do sedaj podprta predvsem v namiznem okolju in se niso tako uveljavila za širšo uporabo. Zaradi napredka v razvoju mobilnih naprav in brezžične komunikacije, je danes v uporabi vedno več mobilnih naprav z različnimi zmogljivostmi, ki učencem ponujajo možnost za lažje učenje in povečanje produktivnosti v šoli. Zaradi tega razloga smo razvili orodje za predstavitev fizikalnih simulacij v obliki mobilne aplikacije xyzMobile. Aplikacija je podprta s strani različnih naprav, tudi pametnih telefonov in tablic. Deluje na različnih velikostih zaslona in izkorišča prednosti prenosljivosti in računskih zmožnosti naprav. Z uporabo okolja jQuery Mobile, ki je prilagojen za naprave na dotik, smo dosegli, da aplikacija ni uporabna le v namiznem okolju, ampak dostopna tudi na drugih napravah. Razvili smo tudi podporo za hkratno manipuliranje simulacije s strani več uporabnikov v realnem času. V ta namen smo implementirali spletni vtičnik, ki omogoča interaktivno komunikacijsko sejo med strežnikom in brskalnikom, kar nam omogoča hkratno kreiranje in manipuliranje simulacije na več napravah.

Ključne besede: simulacije, mobilne aplikacije, jQuery Mobile, spletni vtičnik.

Chapter 1

Introduction

Experiments are a real-world learning experiences that have a positive impact on the learning process, adversely affecting learner motivation in particular. The benefits of real-world situated learning can be brought to the traditional classrooms using educational simulations. Simulations motivate the learner to engage in problem solving, hypothesis testing, experiential learning and development of mental models [1]. They are based on an internal model of a real-world system or phenomena in which some elements have been simplified or omitted in order to facilitate learning [4]. Computer simulations are mimicking (step by step) the anticipated behavior in the natural environment based on the developed mathematical model of physical phenomenon. The changing state of mathematical model is presented to the user, who can monitor it as he would have in the natural environment. In this case, the user gains more control, as he would by conducting the real experiment, because he can slow down, accelerate or even stop the execution whenever he wants.

Beside all of the significant benefits simulations offer in the daily research work and in traditional classrooms, they still haven't achieved the deserved widespread adoption. One of the biggest problems for this is that even with the multitude of devices, most educational simulations still only target desktop environments. Because of this reason, we have created the application xyzMobile, which is a tool for setting up and visualizing physics simulations

that is supported not only on desktop environments, but also on mobile and tablet devices. It is a remake of the program jsXYZ, which is built upon the JavaXYZ program. That means that most of the experiments prepared by original jsXYZ and JavaXYZ applications can be run and visualized by the new xyzMobile application.

The original jsXZY application has been built as a web application, using obsolete technologies. As such, it could only be accessed via desktop clients and has not been designed using contemporary web- and mobile-based technologies. The main purpose of my thesis has therefore been (1) to make this application reachable from as many devices as possible, including mobile devices and tablets, (2) to provide an optimal viewing experience—easy reading and navigation with a minimum of resizing, panning, and scrolling—across a wide range of devices and (3) to support multiuser physical simulations on mobile devices. For this purpose we have comprehensively re-architected the whole design leveraging contemporary technologies, such as jQuery Mobile (1)(2) and WebSockets (3). The jQuery Mobile provides a touch-friendly UI (User Interface) widgets that are specially styled for mobile and tablet devices, whereas WebSockets provide a better interaction between a browser and xyzMobile application, facilitating live content and the creation of real-time simulations.

Chapter 2

Overview of JavaXYZ

2.1 Background

One of the important authoring tools, which supports construction of simulations of natural phenomena was xyZET. It was developed at IPN Kiel (Germany) and has received some excellence awards. Besides this, some additional platform independent applets like xyzViewer and JxyZET were developed. xyZET is a powerful simulation tool written in C, which covers the basic concepts of mechanics and electricity. It is an interactive, graphically oriented tool that permits the presentation of objects and structures in 2D or 3D space. The basic building elements of these objects are particles that are defined by their mass, charge, initial position and velocity. These particles can be connected by springs allowing presentation of complex, non-rigid bodies. Various internal and external forces may be imposed on the particles in the system. Their behavior can be observed during the animation. The basic phenomena from the domain of mechanics and electricity can be explored such as kinematics, conservation of energy and momentum, interacting electric charges, Hook's law and gravity. Other monitors allow displays of force field lines and equipotential planes.

The conceptual learning of the particular phenomena can be achieved by incremental building of the first simple and then increasingly more complex

bodies and structures. These can then be endowed, with various physical parameters. The teaching scenario can be included in accompanying and interacting hypertext. In such ways complete courses in the domains of mechanics and electricity have been created.

Figure 2.1 displays a screenshot of the Java applet of JxyZET, which shows experiment comparing a mathematical and a physical pendulum. The 3D world with the experiment is visualized in one window. Below, in Figure 2.4 is shown the accompanying control frame and the hypertext tutorial. The parameters of the experiment and simulation itself can be controlled by means of controls that are embedded in the hypertext. Most of these simulations are based on the use of Java and JavaScript, and are therefore suitable for implementation on web servers as well as the increasingly popular online classrooms.

The user can change the viewpoint into the visualized world. He or she can activate numerical and graphical monitors, which display some chosen variables. Graphical monitors permit the observation of time dependent variables such as velocities, forces or accelerations of selected objects. The numerical monitors permit the modification of some significant properties of the selected objects.

The gallery of several hundred experiments includes some fairly complex examples with bodies consisting of more than 100 interconnected particles. Most of the experiments are integrated within the tutorials covering mechanics, resonance and electricity.

xyZET was later rewritten in Java and published under the name JavaXYZ. JavaXYZ is in the form of a jar file named JavaXYZ.jar and can be opened as a standalone-application or applet.

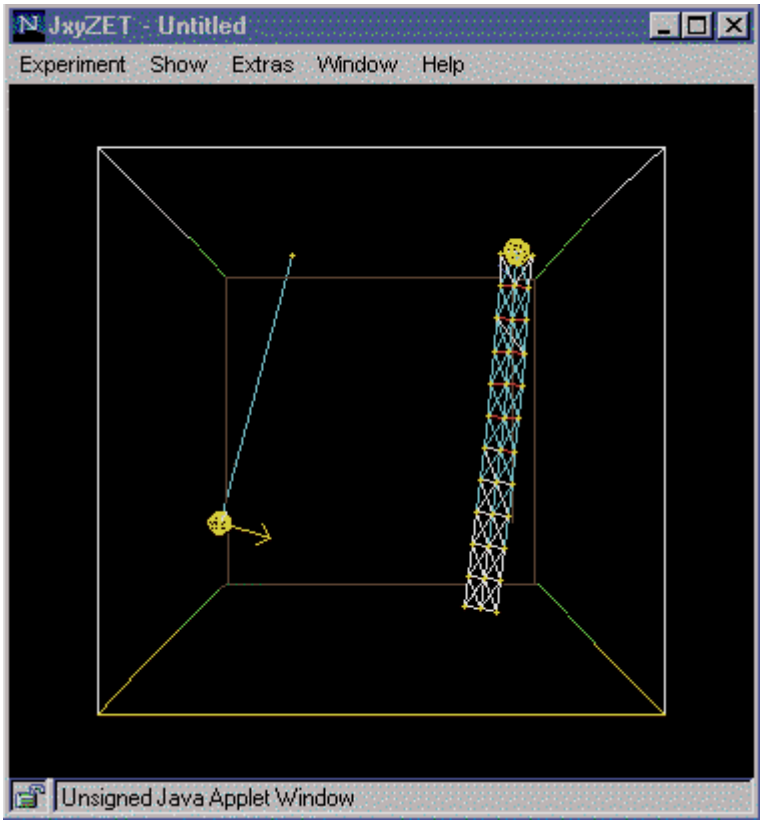

Figure 2.1: Experiment comparing a mathematical and a physical pendulum in JxyZET

JavaXYZ like application represents environment in which we can develop experiments that can be loaded with the applet JavaXYZ. JavaXYZ as an applet can be controlled with JavaScript commands. But, if it is used as an applet, then web page must contain certain necessary elements.

Although the program has been very useful for the preparation of interactive courses, however there are some minor flaws. Among them, probably the most important is the user interface. Since the program opens a separate window, it cannot be completely incorporated into the website, which represent the most appropriate format for courses. Moreover, another problem is the complexity of the interface. Because of the large number of available options, it requires some time for the user to get used to it. Thus, the user

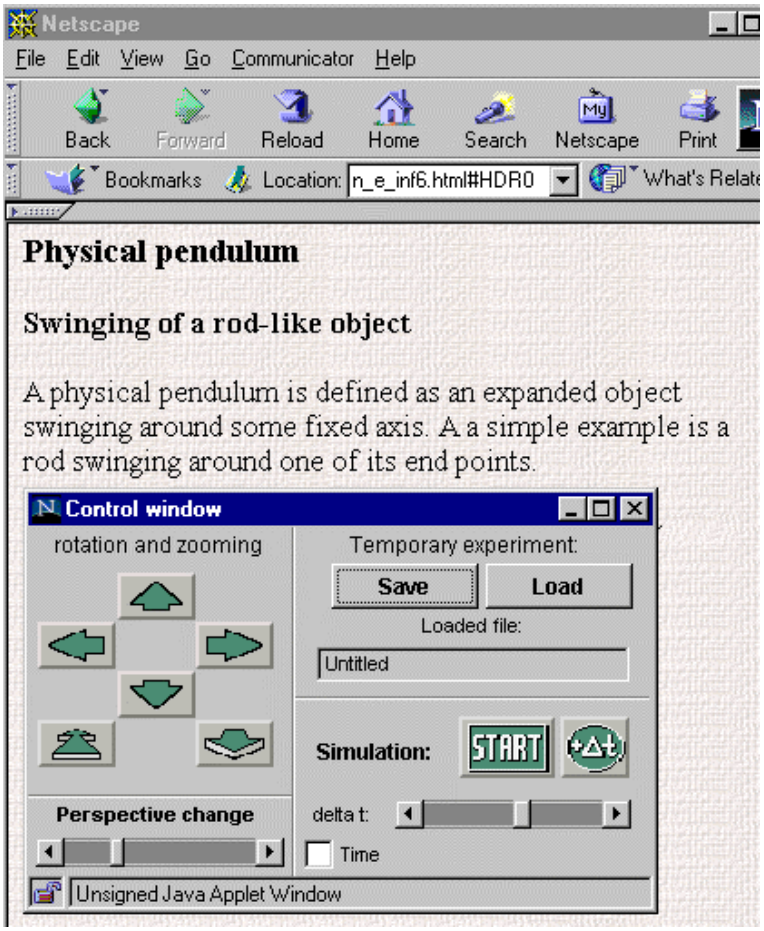

Figure 2.2: Control frame and the hypertext tutorial of JxyZET

attention is unnecessarily diverted from the simulation itself.

2.2 Physical model

For modeling elastic collisions and elastic and non-elastic contact collision forces, equations for conservation of energy and momentum are used. Non-elastic collision simulations are based on Huygens-Newton law, where e is defined as a ratio of a change of momentum between phases of compression and decompression. By manipulating e we can demonstrate all scenarios

between a completely elastic and completely non-elastic collisions.

2.2.1 Elastic forces for modeling spring action

Hooke's law says that the spring elongation (increment in L) is:

$$E = k_0 + k \cdot W \tag{2.1}$$

with k_0 a measurement error and k a parameter expressing the elasticity of the spring. This parameter increases as L increases according to the law:

$$k = k_1 \cdot L \tag{2.2}$$

with k_1 a parameter representing the effects on elasticity of all factors other than L.

Equations (2.1) and (2.2) imply:

$$E = k_0 + k_1 \cdot L \cdot W \tag{2.3}$$

Hooke's law is the basis of sprint action simulation. For example, simulation of rubber band is possible (pulling if l > wavelength) or a shock absorber (pushing if l < wavelength). The effect of these and every other implemented forces on particle velocity and acceleration inside an XYZ cube is given with a numerical solution for Newton's second law.

2.2.2 Damping forces

Real-world springs don't oscillate indefinitely. Non-dampened oscillation is an idealization because real-world systems also include friction and air resistance effects. Because of such energy losses, generally called damping, the amplitude of oscillation gradually decreases and the spring eventually stops moving.

To model a more realistic scenario of dampened spring the force used on is decreased by a factor in every time step, the factor being controlled by a user.

2.2.3 Friction forces

In modeling particle movement inside a cube under the effect of friction force (air resistance for example) the sum of every applied force for every particle is decreased by a force in an inverse direction and proportional to a square of velocity.

The force used for a particle moving along a cube surface points in an inverse direction and is proportional to particle velocity.

2.2.4 Gravitation forces (external and reciprocal)

An external attractive force can be used, proportional to particle mass and directed in a specific direction (-z direction, pointing downwards in the default cube view). Attractive forces simulation is based on general law of attraction.

2.2.5 Coulomb forces (external and reciprocal)

An electrical field (constant or varying in time) that effects charged particles can be established. The field can be established in any direction by setting the value of x, y, and z components.

Simulation of attractive and reflective forces on charged particles is based on Coulomb's law, which states that: *The magnitude of the electrostatic force of interaction between two point charges is directly proportional to the scalar multiplication of the magnitudes of charges and inversely proportional to the square of the distance between them.* The force is along the straight line joining them. If the two charges have the same sign, the electrostatic force between them is repulsive; if they have different sign, the force between them is attractive [2].

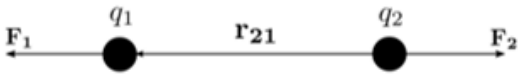

Figure 2.3: Coulomb's law

where q_1 and q_2 are the signed magnitudes of the charges, the scalar r is is the distance between the charges, the vector $r_{21} = r_1 - r_2$ is the vectorial distance between the charges.

2.2.6 Magnetic forces

A continuously constant magnetic field can be established in any direction by setting the value of x, y and z components. In addition, non-homogenous magnetic fields with fixed pole locations and varying intensity can be simulated. The simulation of moving charged particles interaction is based on Lorentz's equation for force.

2.3 Base algorithm

2.3.1 Sequence of commands in the step of calculating

For each particle, we calculate the force caused by all of the other particles.

$$f = mutGravConst \cdot \frac{m_1 \cdot m_2}{r_2} \tag{2.4}$$

$$f+ = -coulConst \cdot \frac{q_1 \cdot q_2}{r_2} \tag{2.5}$$

For each particle connected to a spring, spring force is respectively calculated.

$$f = (r - mylen) \cdot mycon \tag{2.6}$$

where r = actual length, mylen= rest length and myCon= spring constant

The force caused by an external electric field is calculated as

$$f+ = q_1 \cdot E \tag{2.7}$$

The force caused by an external field of attraction is calculated as

$$f+ = m_1 \cdot G \tag{2.8}$$

Lorentz's force with respect to the speed of the particle is calculated as

$$f+ = q_1 \cdot (v \times B) \tag{2.9}$$

The acceleration of the particle is calculated as

$$a = \frac{f}{m_1} \tag{2.10}$$

In the last two steps we calculate the new speed and if predicted, we calculate the friction factor.

2.4 Mathematical basics

Mapping a mathematical model to a picture on a screen is done in three steps which make the otherwise complicated procedure more controllable and make it more structured so that the program code, which executes the procedure, isn't overly complicated. Every calculation is based on a spatial clockwise Cartesian coordinate system, shown on Figure 2.4. The coordinate system introduces a measuring system, crucial for mathematical modeling of any sorts.

Every object in space is initially positioned in its own coordinate system. Inside, the object's shape is precisely defined. Such object descriptions based on local coordinate systems enables simple use of the same mathematical description regardless of where in space the object is located. When modeling objects we can therefore completely ignore the complexity introduced by the ability to move and rotate objects freely in space, and instead focus only on the object's shape in its initial position.

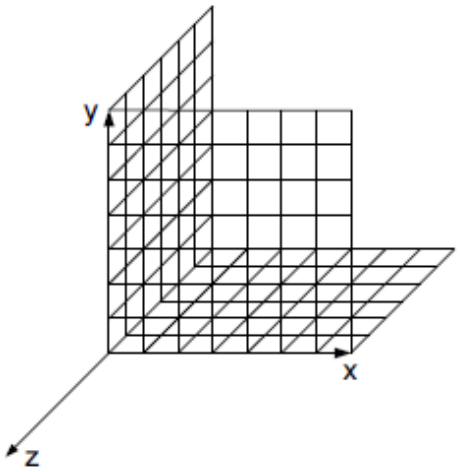

Figure 2.4: Spatial clockwise Cartesian coordinate system

The first step from such object in its initial position to its final image on a screen is a transformation of its coordinates from local to global coordinate system (world coordinate system). In doing this, the entire virtual now only has one origin, with respect to which the position and rotation of every object is expressed. This way, we move from individual object description to a unified description of the entire space. Because the user will not observe the space from a single point – the coordinate system of this space – we need an additional transformation with which the origin is moved to camera location. This step results in every object being expressed in camera (observer) coordinate system meaning very easy determination of object positions with respect to camera point (from where the user observes).

Only the final step remains to finally render the objects on a screen. The three-dimensional coordinates from camera coordinate system need to be projected on a two-dimensional surface which will be rendered on a screen. In contrast to previous transformations, this is a perspective transformation and serves to realistically display the calculated space model on a screen with respect to camera (observer) parameters.

The transformations mentioned are very similar to each other and are

composed of four different components:

- Translation component which represents a simple parallel translation of a coordinate system (and therefore changes the object's position).
- Rotation component which rotates the coordinate system (and therefore changes the object's rotation).
- Scaling component enables object scaling (and is therefore used for transformation between local and world coordinate systems in order to set objects' scales).
- Perspective component.

Chapter 3

Simulation tool jsXYZ

In time several problems occurred by using Java Applets, which were embed into web pages. Interactivity of web tutorials has become hindered by security warnings, as depicted in the Figure 3.1 below.

Figure 3.1: Security warning

With the similar problems we also encounter in hypertext notebooks, which use the Java-XYZ applets. For this reason, a new reengineering of the source code and a migration to a HTML and JavaScript technologies has been performed. The new tool is available under the name jsXYZ and supports most of already developed experiments. One notable issue has been the transfer from strictly object oriented Java environments into a scripting language JavaScript. The model of physical engine has thereby been preserved (simulation of different forces). The tool three.js significantly simplifies the graphical 3D visualization and is described in a separate section below.

In such way, jsXYZ enables interactive view of pre-prepared simulations, but does not enable the preparation of new simulations. For this purpose we can still use the JavaXYZ as a useful tool, because Java applications do not have such security issues as perceived in previous Java Applets. Experiments, which we develop and save using the initial version of JavaXYZ can later on be imported to jsXYZ. Both environments present an application as a whole.

3.1 JavaScript

JavaScript is a lightweight, interpreted programming language with object-oriented capabilities that allows you to build interactivity into otherwise static HTML pages. It started life as LiveScript, but Netscape changed the name, possibly because of the excitement being generated by Java to JavaScript. JavaScript made its first appearance in Netscape 2.0 in 1995 with a name LiveScript [3].

The general-purpose core of the language has been embedded in almost all popular web browsers. JavaScript as a programming language is designed for creating network-centric applications. It is an open and cross-platform, complementary and integrated with Java and HTML.

The most common form of the language is the client-side JavaScript. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser. It means that a web page no longer needs

to be static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

The JavaScript client-side mechanism features many advantages over traditional CGI (Common Gateway Interface) server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field. The JavaScript code is executed when the user submits the form, and only if all the entries are valid they would be submitted to the Web Server.

Moreover, JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user explicitly or implicitly initiates. In fact, there are many merits of using JavaScript, which are making it one of the most popular programming languages. Some of them are listed below:

- Less server interaction - You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.

- Immediate feedback to the visitors - They don't have to wait for a page reload to see if they have forgotten to enter something.

- Increased interactivity - You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.

- Richer interfaces - You can use JavaScript to include such items as drag-and-drop components and sliders to give a rich interface to your site visitors.

Despite all the advantages, advanced JavaScript programming (especially the complex handling of browser differences) can often be very difficult and time-consuming to work with. To deal with these difficulties, a lot of JavaScript (helper) libraries have been developed.

3.2 Three.js

Three.js is one type of such JavaScript (helper) libraries that is aimed at rendering computer generated 3D graphics in a web browser. As a high-level library it makes possible to author complex 3D computer animations that display in the browser without the effort required for a traditional standalone application or a plugin [4]. Three.js abstracts away the lower-level API (Application Programming Interface) calls and by this way it makes developing WebGL (Web Graphics Library) applications easier and more productive. While a simple cube in raw WebGL would turn out hundreds of lines of JavaScript and shader code, a three.js equivalent is only a fraction of that. With this framework, it is not necessary to explicitly compile and apply shaders with WebGL commands. All of that is done for you in the background.

Furthermore, three.js features different renderers: Canvas, SVG (Scalable Vector Graphics) and WebGL. The WebGL renderer has the ability to render GPU (Graphics Processing Unit) accelerated 3D graphics, which means that it can utilize the processing power of the graphics card to draw 3D scenes. The three.js library simplifies the process of setting up a rendering environment greatly, by offering features such as matrix mathematics, scene-graph, 3D model import, and support for several common shader effects like normal-, specular- and shadow-mapping. WebGL renderer has way better performance than CanvasRenderer. The Canvas renderer, on the other side draws the scenes using the (slower) Canvas 2D Context API. This renderer can be a nice fallback from WebGLRenderer for simple scenes. Both of the renderers are embedded in the web page using an HTML5 canvas tag [5].

In addition, three.js offers many other different features. Among them are effects (anaglyph, cross-eyed and parallax barrier), lights (ambient, direction, point and spot lights), shadows (cast and receive), animation (armatures, forward kinematics, inverse kinematics, morph, keyframe) and many other [6].

It was created by Ricardo Cabello Miguel and the first version was re-

leased in 2010 [7]. It is currently maintained as an open source project in a repository on GitHub, allowing the library to be reused within proprietary software as long as a copy of the license is included in the software [8].

Chapter 4

jQuery Mobile

jQuery Mobile is a client side framework optimized for touch devices. It is an open source project sponsored by large mobile and media companies. It offers a very modern and highly customizable UI. The user interface is based on HTML5, CSS3 (Cascading Style Sheets), and the very popular jQuery JavaScript library, and uses declarative coding, making it easy to use and learn. It provides touch-friendly UI widgets that are specially styled for mobile devices. jQuery Mobile has a powerful theming framework to style applications [9]. It supports AJAX for various tasks, such as page navigation and transitions.

As jQuery Mobile follows the open web standards, it is compatible with a wide range of browsers and platforms. Application written in jQuery mobile will work seamlessly on iPhones, iPads, Android phones and tablets, Blackberry, Bada, Windows, Symbian, Meego, and even the upcoming HTML5-based platforms, such as Boot2Gecko and Tizen. The same code will run on Chrome, Firefox, Opera, IE, Safari, and other browsers on your desktop. Further, it will work even on smart TVs or any other gadget that has a compatible browser which is compliant with the open web standards. Considering the numerous browsers and platforms that it supports its market potential is phenomenal [10].

4.1 CSS Framework

jQuery Mobile offers CSS-based enhancements for common user interface elements, such as:

- CSS classes for common styles.
- Multi-column layout grids.
- Responsive layout grids.
- jQuery Mobile theme.

4.2 Events

With jQuery Mobile there are also offered several custom events that build upon native events to create useful hooks for development. Some of them are for example *hashchange* that enables bookmarkable hash history, *mobileinit* event, which indicates that jQuery Mobile has finished loading, or *pagecreate* that is triggered when the page has been created in the DOM (via ajax or other) and after all widgets have had an opportunity to enhance the contained markup.

4.3 Icons

In addition, jQuery Mobile provides a set of built-in icons that can be applied to buttons, collapsibles, listview buttons and more. It offers a number of icons that can be used by applying a *data-icon* attribute or an *ui-icon* class to a suitable widget. There is an SVG and PNG (Portable Network Graphics) image of each icon. By default the SVG icons, that look great on both SD (Standard Definition) and HD (High Definition) screens, are used. On platforms that don't support SVG the framework falls back to PNG icons. The icon name is self-describing. For example, the following will display a button with a home icon (Listing 4.1).

```
<a href="index.html" class="ui-btn ui-icon-home ui-btn-icon-left">Home</a>
```

Listing 4.1: Button with a home icon

4.4 Methods

jQuery Mobile also exposes several methods on the $.mobile object for use in the applications. For example *jqmHijackable()* is intended for users that wish to respect data-ajax=false parent elements during custom form and link binding jQuery Mobile provides the $.fn.jqmHijackable filter method. Another example is an utility method *path.parseUrl()* for parsing an URL (Uniform Resource Locator) and its relative variants into an object that makes accessing the components of the URL easy. There is also a collection of methods for dealing with paths, such as *path.get()* method for determining the directory portion of an URL and *path.makePathAbsolute()* for converting a relative file or directory path into an absolute path.

4.5 Reference

The jQuery Mobile framework uses HTML5 (Hyper Text Markup Language) data-attributes. Data-attributes allow for markup-based initialization and configuration of widgets. These attributes are completely optional; calling plugins manually and passing options directly is also supported. To avoid naming conflicts with other plugins or frameworks that also use data-attributes, user should just set a custom namespace by modifying the ns global option.

Furthermore, unlike other jQuery projects, such as jQuery and jQuery UI, jQuery Mobile automatically applies many markup enhancements as soon as it loads (long before the document.ready event fires). These enhancements are applied based on jQuery Mobile's default settings, which are designed to work with common scenarios. If changes to the settings are needed, they are

easy to configure. When jQuery Mobile starts, it triggers a mobileinit event on the document object. To override default settings,the user should bind to mobileinit, as shown in the Listing 4.2.

```
$( document ).on( "mobileinit", function() {
  //apply overrides here
});
```

Listing 4.2: Mobileinit event

Because the mobileinit event is triggered immediately, it is needed to bind the event handler before jQuery Mobile is loaded. Linking to the JavaScript files should be done in the following order (Listing 4.3):

```
<script src="jquery.js"></script>
<script src="custom-scripting.js"></script>
<script src="jquery-mobile.js"></script>
```

Listing 4.3: Linking to JavaScript files

Moreover, jQuery Mobile is offering a way to override the default settings. It can be done by extending the $.mobile object using jQuery's $.extend method (Listing 4.4).

```
$( document ).on( "mobileinit", function() {
  $.extend( $.mobile , {
    foo: bar
  });
});
```

Listing 4.4: Overriding the default settings using extend method

Alternatively, they can be changed using object property notation, as presented below (Listing 4.5).

```
$( document ).on( "mobileinit", function() {
  $.mobile.foo = bar;
});
```

Listing 4.5: Overriding the default settings using object property notation

4.6 Widgets

Widgets are feature-rich, stateful plugins that have a full life-cycle, along with methods and events. The most important widgets, which are used in our mobile application will be described in the following subsections.

4.6.1 Button Widget

Creates a button widget. Buttons are coded with standard HTML input elements, then enhanced by jQuery Mobile to make them more attractive and useable on a mobile device.

For easier styling, the framework automatically converts any input element with a type of submit, reset, or button into a custom styled button. Because of that reason there is no need to add the data-role="button" attribute. However, if needed, the button plugin on any selector can be called directly, just like any jQuery plugin (Listing 4.6).

```
$( "[type='submit']" ).button();
```

Listing 4.6: Form button

Example of one button widget, used in xyzMobile application is shown in Figure 4.1.

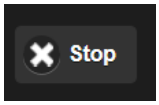

Figure 4.1: Button widget

4.6.2 Checkboxradio Widget

Creates a checkboxradio widget. Checkboxes are used to provide a list of options where more than one can be selected. Traditional desktop checkboxes

are not optimized for touch input so in jQuery Mobile, the label is styled for the checkboxes so they are larger and look clickable. A custom set of icons are added to the label to provide additional visual feedback.

A single checkbox can be created, with adding an input with a type = "checkbox" attribute and a corresponding label. If the input isn't wrapped in its corresponding label, the for attribute of the label should be set to match the id of the input so they are semantically associated.

Furthermore, checkboxes can be grouped vertically or as a horizontal toggle sets. Vertically grouped checkboxes are needed in cases for example where multiple checkboxes are listed under a question title. To visually integrate multiple checkboxes into a grouped button set, the framework will automatically remove all margins between buttons. Checkboxes can also be used for grouped button sets where more than one button can be selected at once, such as the bold, italic and underline button group seen in word processors. That's an example of horizontal toggle sets. To make a horizontal button set, add the data-type="horizontal" to the fieldset as shown in Listing 4.7.

```
<fieldset data-role="controlgroup" data-type="horizontal">
  <label>Axes<input type="checkbox" checked="true"></label>
  <label>Cube<input type="checkbox" checked="true"></label>
  <label>Floor<input type="checkbox" checked="true"></label>
</fieldset>
```

Listing 4.7: Horizontal button set

Below, in Figure 4.2 is presented the look of the horizontal button set used in our mobile application.

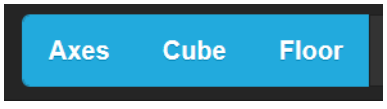

Figure 4.2: Horizontal button set

4.6.3 Radio buttons

Radio buttons are used to provide a list of options where only a single item can be selected. Traditional desktop radio buttons are not optimized for touch input so jQuery Mobile styles the label for the radio buttons so they are larger and look clickable.

A set of radio buttons is created, with adding an input with a type="radio" attribute and a corresponding label. "For" attribute of the label should be set to match the id of the input. In this way they are semantically associated, as shown in the code in the Listing 4.8

```
<form>
  <label for="radio-choice-perspective">Perspective camera</label>
  <input name="radio-choice-0" id="radio-choice-perspective" type="radio">
  <label for="radio-choice-orthogonal">Orthographic camera </label>
  <input name="radio-choice-0" id="radio-choice-orthogonal" type="radio">
</form>
```

Listing 4.8: Set of radio buttons

This will produce a set of two radio buttons with their corresponding labels (Figure 4.3).

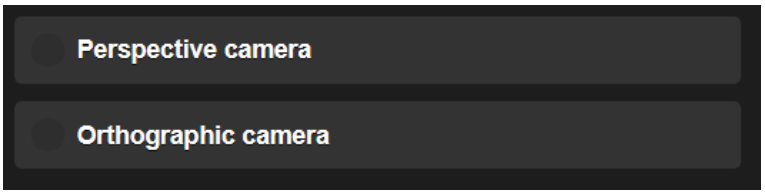

Figure 4.3: Set of two radio buttons

4.6.4 Collapsibleset Widget

Creates a collapsible block of content. For creating a collapsible block of content, user just has to create a container and add the data-role="collapsible"

attribute. Then, using the data-content-theme attribute he is allowed to set a theme for the content of the collapsible.

Directly inside this container, user can add any header (H1-H6) or legend element. The framework will style the header to look like a clickable button and add an icon to the left to indicate it's expandable.

After the header, it is allowed to add any HTML markup we want to be collapsible. The framework will wrap this markup in a container that will be hidden/shown when the heading is clicked.

The default icon of collapsible headings can be overridden by using the data-collapsed-icon and data-expanded-icon attributes.

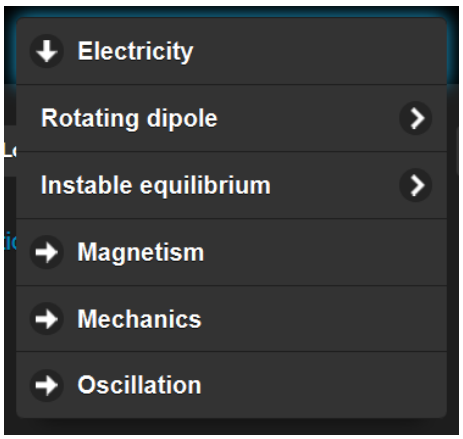

Figure 4.4: Collapsibleset widget

The example in Figure 4.4 uses data-collapsed-icon="arrow-r" and data-expanded-icon="arrow-d". Below, in the Listing 4.9 is presented only a part of the original source code for creating collapsibleset widget in our mobile application xyzMobile.

```
<div data-role="collapsibleset" data-theme="b" data-content-theme="a"
data-collapsed-icon="arrow-r" data-expanded-icon="arrow-d">
  <div data-role="collapsible" data-inset="false">
    <h2>Electricity</h2>
      <ul data-role="listview">
        <li><a data-rel="dialog">Rotating dipole</a></li>
        <li><a data-rel="dialog">Instable equilibrium</a></li>
      </ul>
  </div>
</div>
```

Listing 4.9: Collapsibleset widget

4.6.5 Controlgroup Widget

This widget groups buttons together. Occasionally, user may want to visually group a set of buttons to form a single block that looks contained like a navigation component. To get this effect, he just needs to wrap a set of buttons in a container with the data-role="controlgroup" attribute (Listing 4.10). The framework will create a vertical button group, remove all margins and drop shadows between the buttons, and only round the first and last buttons of the set to create the effect that they are grouped together (Figure 4.5).

Figure 4.5: Vertical button group

```
<div data-role="controlgroup">
  <a href="#" class="ui-btn ui-corner-all">Yes</a>
  <a href="#" class="ui-btn ui-corner-all">No</a>
  <a href="#" class="ui-btn ui-corner-all">Maybe</a>
</div>
```

Listing 4.10: Vertical button group

Moreover, by adding the data-type="horizontal" attribute to the controlgroup container, user can swap to a horizontal-style group that floats the buttons side-by-side and sets the width to only be large enough to fit the content. Example of controlgroup widget with horizontally positioned checkbox buttons was already presented in Figure 4.2.

4.6.6 Dialog Widget

Dialog widget opens content in an interactive overlay. Any page can be presented as a modal dialog by adding the data-rel="dialog" attribute to the page anchor link, as shown in the Listing 4.11. When the "dialog" attribute is applied, the framework adds styles to add rounded corners, margins around the page and a dark background to make the "dialog" appear to be suspended above the page.

```
<a href="foo.html" data-rel="dialog">Open dialog</a>
```

Listing 4.11: Dialog widget

Typically, users are accustomed to see dialog when they are warned about an action they are performing, which may have undesirable outcome for them. One example of this kind of dialog widget is presented below, Figure 4.6.

Another use of the dialog was presented in Figure 4.4 in the section of collapsibleset widget.

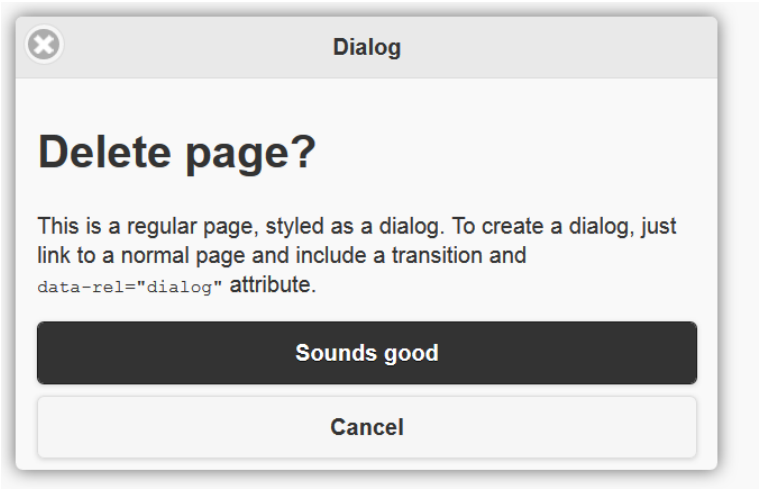

Figure 4.6: Dialog widget

4.6.7 Listview Widget

Creates a listview widget. A listview is coded as a simple unordered list containing linked list items with a data-role="listview" attribute. The list is transformed into a mobile-friendly listview with right arrow indicator that fills the full width of the browser window. When user tap on the list item, the framework will trigger a click on the first link inside the list item, issue an Ajax request for the URL in the link, create the new page in the DOM (Document Object Model), then kick off a page transition.

Here in Listing 4.12 is the HTML markup for a list used in this mobile application.

```
<ul data-role="listview">
  <li><a data-rel="dialog">Rotating dipole</a></li>
  <li><a data-rel="dialog">Instable equilibrium</a></li>
</ul>
```

Listing 4.12: Listview widget

This results in the following figure (Figure 4.7):

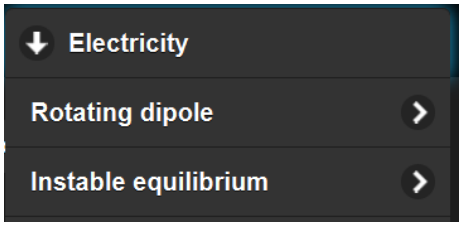

Figure 4.7: Listview widget

4.6.8 Navbar Widget

This creates a navbar widget. jQuery Mobile has a very basic navbar widget that is useful for providing up to 5 buttons with optional icons in a bar, typically within a header or footer. There is also a persistent navbar variation that works more like a tab bar that stays fixed as you navigate across pages.

A navbar is coded as an unordered list of links wrapped in a container element that has the data-role="navbar" attribute. When a link in the navbar is clicked it gets the active (selected) state. To set an item to the active state upon initialization of the navbar, user just need to add class="ui-btn-active" to the corresponding anchor in his markup.

Furthermore, by adding the data-icon attribute, users can add icons to the navbar items, specifying a standard mobile icon to each anchor.

Listing 4.13 is an example of a two-button navbar with custom icon.

```
<div data-role="navbar"><ul>
 <li>
  <a href="#" data-role="tab" data-icon="grid" class="ui-btn-active">Display commands</a>
 </li><li>
  <a href="#" data-role="tab" data-icon="grid">Photo slider</a>
 </li>
</ul></div>
```

Listing 4.13: Navbar widget

The navbar items are set to divide the space evenly so in this case, each button is 1/2 the width of the browser window, as shown in the Figure 4.8

Figure 4.8: Navbar widget

4.6.9 Page Widget

The page widget is responsible for managing a single item in jQuery Mobile's page-based architecture. It is designed to support either single page widgets within a HTML document, or multiple local internal linked page widgets within a HTML document. The example below illustrates two page widgets in one single HTML document.

```
<div id="page-1" data-role="page" data-theme="b" >
  %content in the first page goes here
</div>
<div id="page-2" data-role="page" data-theme="b" >
  %content in the second page goes here
</div>
```

Listing 4.14: Page widget

4.6.10 Popup Widget

Opens content in a popup. For creating a popup, it is needed just to add the data-role="popup" attribute to a div with the popup contents.

The popup consists of two elements: the screen, which is a transparent or translucent element that covers the entire document, and the container, which is the popup itself. One example of popup was already shown in the section of collapsibleset widget. Another example is shown below in Listing 4.15

```
<a href="#popupBasic" data-rel="popup">Open Popup</a>
<div data-role="popup" id="popupBasic">
  <p>This is a completely basic popup, no options set.</p>
</div>
```

Listing 4.15: Popup widget

This is in fact a very simple and basic popup. Its look is presented on Figures 4.9 and 4.10 respectively.

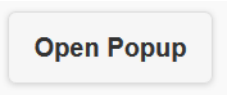

Figure 4.9: Popup widget before opening

By default, popups have no transition to make them open as quickly as possible. To set the transition used for a popup, user needs to add the data-transition attribute to the link that references the popup. The reverse version of the transition will be used when closing the popup.

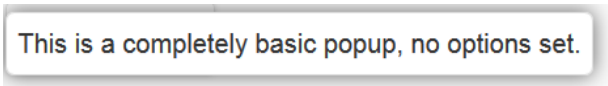

Figure 4.10: Content of the popup

For performance reasons on mobile devices, it is recommended to use simpler transitions like pop, fade or none for smooth and fast popup animations, especially with larger or complex widgets within a popup. Because by default, devices that lack 3D support (such as Android 2.x) will fallback to "fade" for all transition types.

4.6.11 Slider Widget

To add a slider widget to the page, standard input with the type="range" attribute should be used. The input's value is used to configure the starting position of the handle and the value is populated in the text input. Furthermore to set the slider's range user just need to specify min and max attribute values.

In the example of Listing 4.16 the acceptable range is from -157 to 157. The value attribute is used to define the initial value. Moreover, "for" attribute of the label is set to match the id of the input so they are semantically associated.

```
<form>
  <label for="sliderZ">Rotation around z axis</label>
  <input id="sliderZ" data-track-theme="a"
  data-theme="a" min="-157" max="157" value="0" type="range" >
</form>
```

Listing 4.16: Slider widget

Slider with these settings is displayed in Figure 4.11.

Figure 4.11: Slider widget

In case, the user wants to constrain input to specific increments, he just needs to add the step attribute.

4.6.12 Toolbar Widget

Toolbar widget adds toolbars to the top and/or bottom of the page. Headers and footers are elements that precede or succeed the page content. The toolbar widget allows users to create headers and footers.

The header bar that serves as the page title, is usually the first element inside each mobile page, and typically contains a page title and up to two buttons. In fact, it is a toolbar at the top of the page that usually contains the page title text and optional buttons positioned to the left and/or right of the title for navigation or actions. Headers can optionally be positioned as fixed so they remain at the top of the screen at all times instead of scrolling with the page.

In the xyzMobile application the header contains navbar widget, like illustrated in the Figure 4.8. Another simpler version of header, which contains only the page title is presented below.

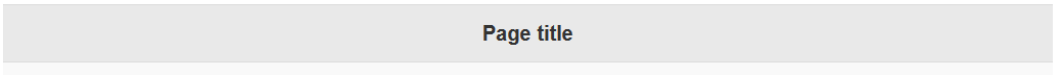

Figure 4.12: Header bar

The title text is normally an H1 heading element, as shown in the Listing 4.17, but it's possible to use any heading level (H1-H6) to allow for semantic flexibility.

```
<div data-role="header">
  <h1> Page Title </h1>
</div>
```

Listing 4.17: Header bar

On the other hand, the footer bar is usually the last element inside each mobile page, and tends to be more freeform than the header in terms of content and functionality, but typically contains a combination of text and buttons. The footer bar has the same basic structure as the header except it uses the data-role attribute value of footer (Listing 4.18).

```
<div data-role="footer">
  <h4>Footer content</h4>
</div>
```

Listing 4.18: Footer bar

This code will produce the following footer (Figure 4.13)

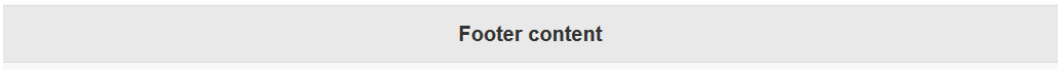

Figure 4.13: Footer bar

Chapter 5

xyzMobile - Adjustment of jsXYZ for mobile devices

So far the jsXZY application has been built as a web application, using obsolete technologies. As such, it could only be accessed via desktop clients and has not been designed using Responsive Web Design principles.

The main purpose of my thesis has been:

- To make this application reachable from as many devices as possible, including mobile devices and tablets.
- To provide an optimal viewing experience—easy reading and navigation with a minimum of resizing, panning, and scrolling—across a wide range of devices.
- To support multiuser physical simulations on portable devices.

For this purpose we have comprehensively re-architected the whole design leveraging contemporary technologies, such as jQuery Mobile. Figure 5.1 represents the design of the user interface for the jsXYZ application.

The interface of the new redesigned application, xyzMobile is shown on Figure 5.2.

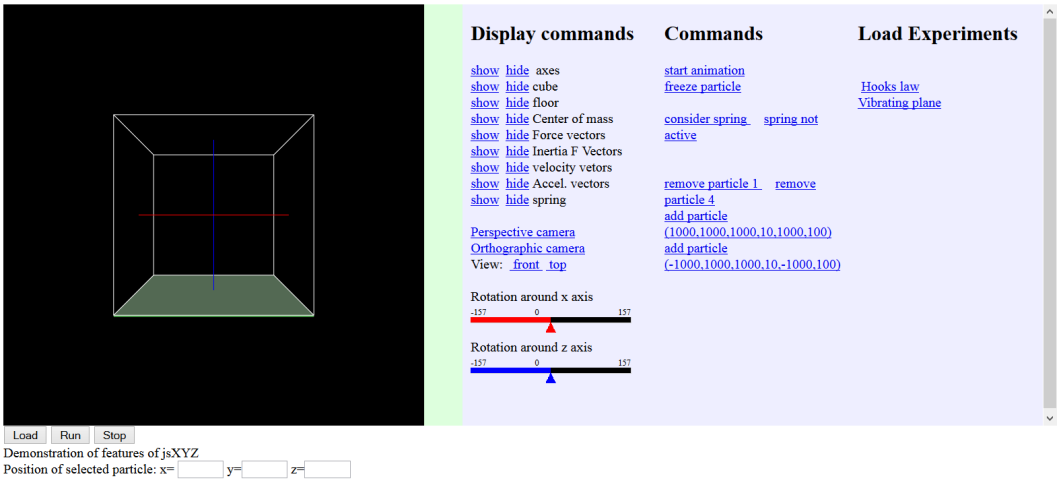

Figure 5.1: User interface of jsXYZ application

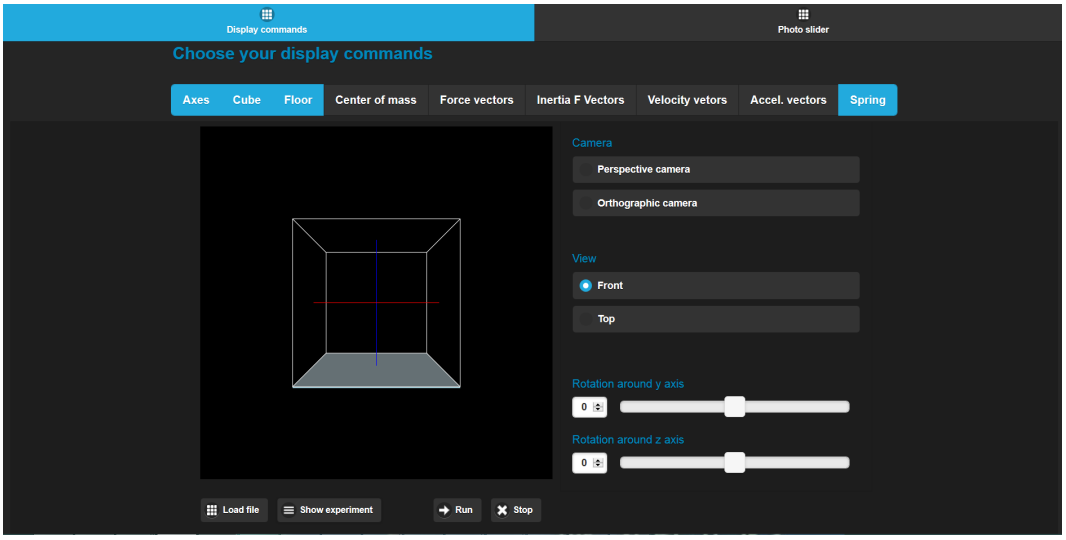

Figure 5.2: User interface of xyzMobile application

5.1 Support for mobile devices and tablets

In order to effectively redesign and alter the look of the whole application, we have used specific jQuery Mobile elements.

Firstly, we started our redesign with display commands. They allow users

to choose between different physical properties of the simulation, like center of mass, force vectors, velocity vectors or to choose how the simulation will be presented, for example: with or without axes, whether the cube will be shown or not, etc. Because xyzMobile is an application that is meant to be a supplement to traditional teaching, display commands are offering students, who represent the majority of end users, the features for interactivity and visualization. In fact, these features are necessary for them to better understand some difficult physical laws. This is making display commands the most important commands, the core of the application.

Despite their importance, in the previous application they have been presented like a vertical list of hyperlinks, making it difficult for the user to differentiate between more and less important features. Because the main function of a good user interface is to provide users with an intuitive mapping between user's intention and application's function, we have decided to align display commands as horizontal commands.

For this purpose, we have used jQuery Mobile element horizontal toggle sets of checkboxes, which visually integrate multiple checkboxes into a grouped button set. Another reason for using checkboxes is because they provide a list of options where more than one can be selected and the user can easily distinguish among selected and unselected option. This is giving the user a clear picture of what is presented to him on the simulation. The source code for horizontal toggle sets of checkboxes was already presented in Listing 4.7

Figure 5.3 shows the comparison between the previous look of display commands (Figure 5.3 - a) and the current, redesigned look (Figure 5.3 - b).

Furthermore, the settings for the camera, view and rotation had to be modified as well. Similar to the display commands, these settings were presented using hyperlinks. They were not grouped and the user had difficulty to understand which commands are tied together. In order to make the content readable, scannable and easy to perceive, content blocks need to be visually separated. In other words, each element needs to be defined and presented

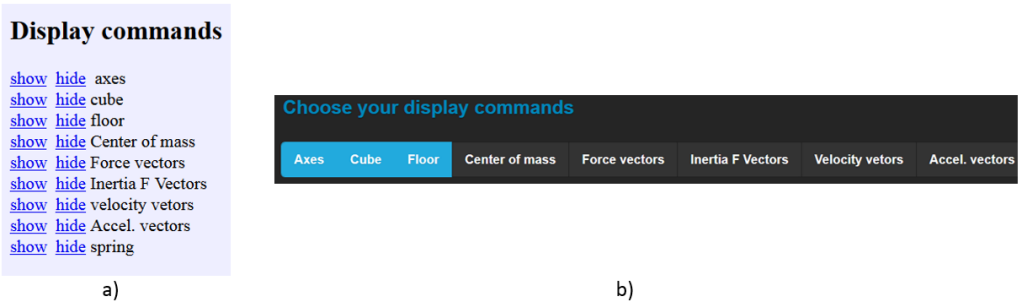

Figure 5.3: Previous (a) and redesigned look (b) of display commands

as a separate element. In fact, the separation of elements in a layout is one of the simplest ways to achieve a cleaner user interface that the user can easily interact with. However, if many elements are visually separated, the interface contains more chunks of information and consequently the layout becomes more complex.

Therefore, to make sure that the layout remains scannable, the visual separation needs to be subtle. Considering this in our application, separation between three the different groups: camera, view and rotation is made by using a vertical blank space. This is providing visual support and is making the user interface friendlier to the user. Figure 5.4 - a shows the look of the camera, view and rotation commands in the jsXYZ application. On the other hand, Figure 5.4 - b presents the modified look in the new application.

Moreover, the camera and view settings are used to change the look of the cube. Camera has two options: perspective and orthographic camera. These are in fact, just different types of projections, ways to transform the cube from one dimensionality to another. An orthographic projection is a very simplistic projection. It is a means of representing a three-dimensional object in two dimensions. It is a form of parallel projection, where all the projection lines are orthogonal to the projection plane, resulting in every plane of the scene appearing in affine transformation on the viewing surface.

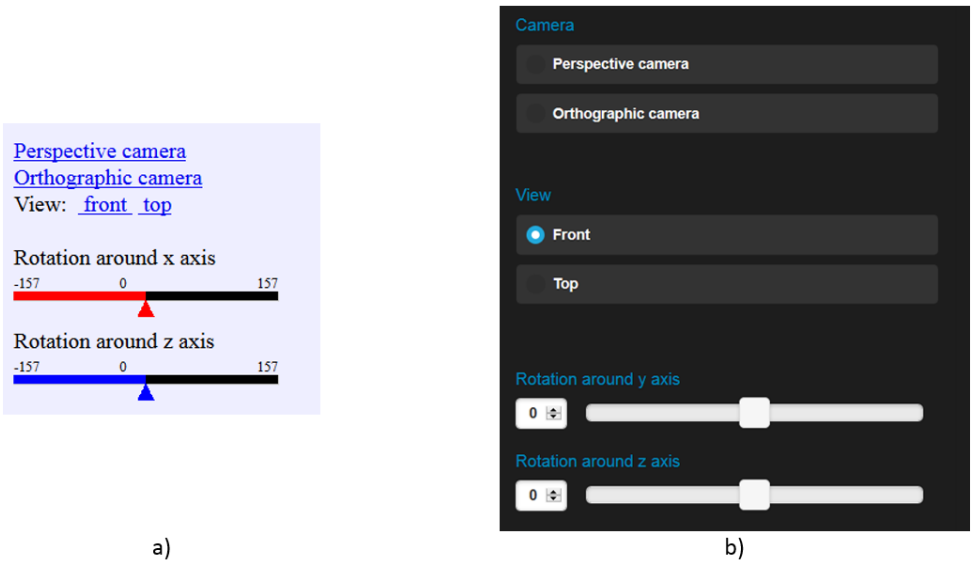

Figure 5.4: Previous and modified look of camera, view and rotation commands

The look of the cube in the orthographic projection is shown in Figure 5.5

Although orthographic projections can be useful, perspective projections create more realistic looking scenes, so that's why the user will most likely be using them more often. While orthographic projection ignores the effect to allow accurate measurements, perspective definition shows distant objects as smaller to provide additional realism. In perspective projections, as an object gets farther from the viewer it will appear smaller on the screen - an effect often referred to as foreshortening. This perspective projection is enabled for the user by clicking the button of perspective camera.

The view setting with its two options: front and top, allow the user to change the view on the cube. With these options user has more different ways of observing the simulations shown within the cube. This is done by changing the respective camera position, for the front view just the z coordinate of the

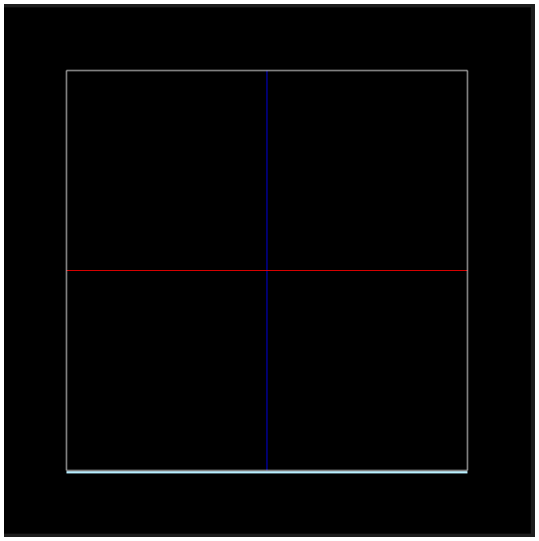

Figure 5.5: Orthographic projection of the cube

camera is changed and for the top the x coordinate. The front view is the default option, enabled when the user opens the application, because it is the most common way of watching the simulations. Top view of the cube is presented in the Figure 5.6. The goal of all of these features is to encourage students to explore the application, so they can see how the simulations are behaving and through their own ideas and experimentation to understand and to learn them.

Features like camera and view, are implemented in the code using jQuery Mobile elements of radio buttons. We chose this type of representation, because radio buttons are used to provide a list of options where only a single item can be selected. After the selection user can easily see what he had chosen and can change his choice later if he wants to. The source code of the camera and view features is shown below (Listing 5.1).

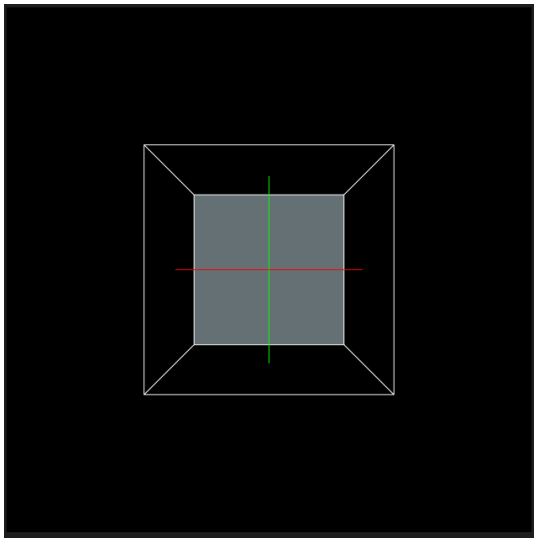

Figure 5.6: Top view of the cube

```
<label>Camera</label>
<form >
  <label for="radio-choice-perspective">Perspective camera</label>
  <input id="radio-choice-perspective" class="perspCamera" type="radio">
  <label for="radio-choice-orthogonal">Orthographic camera </label>
  <input id="radio-choice-orthogonal" class="custom orthoCamera"  type="radio">
</form>

<label>View</label>
<form >
  <label for="radio-choice-front">Label</label>
  <input id="radio-choice-front" type="radio" class="frontView"  checked >
  <label for="radio-choice-top">Top</label>
  <input name="radio-choice-0" id="radio-choice-top" class="custom topView" type="radio">
</form>
```

Listing 5.1: Camera and view commands

Furthermore, the options of rotation around x axis and rotation around y axis in the jsXYZ application was implemented entirely by using JavaScript code. These options were functional, but the design did not provide optimal viewing experience. For that purpose, in xyzMobile the rotation is made by slider widget, with slider range values between -157 to 157 and initial value of 0. In Listing 4.16 in section of slider widget, you can see the source code of the slider. On the other hand, below in the Figure 5.7 is presented the comparison between the old (a) and redesigned look of the rotation features

(b).

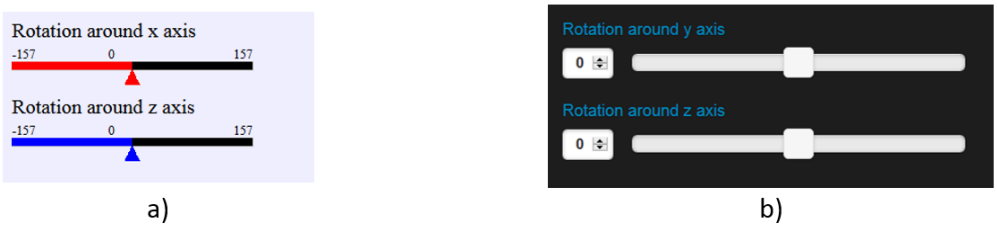

Figure 5.7: Comparison between old (a) and redesigned look (b) of the rotation feature

Additionally, in the second column of the user interface of the jsXYZ application there were commands like start animation, freeze particle, remove or add particles. Because of the large number of available options, the user interface was more complex and more difficult to interact with. One of the main principles of the user interface design, which are intended to improve the quality of the user interface is simplicity principle. It states that the design should make simple, common tasks easy, communicating clearly and simply [11].For this reason, in the xyzMobile application we have decided to reduce the complexity and enhance the information potential of a small set of interface components, with excluding the commands which are not necessary and which do not have great importance for the performance of the simulations, like adding, removing or freezing particles [12].This approach results in expressive design solution, better structure, more usable and more approachable application, for both the novice and experienced users. Simplicity does not mean simplistic solutions, lack of functionality or limited information, but on contrary it presents the ultimate sophistication.

xyzMobile is an application which can be used in many different educational environments, including lecture, individual or small group activities, homework, and lab. While it offers many of the same benefits as doing

demonstrations using real equipment, the application also have several additional advantages. Firstly, it can be used in classrooms where the real equipment is either not available or very difficult to set up. Secondly, it can be used to perform "experiments" that are otherwise impossible to do. Especially, for the second reason the application includes simulations which will help students to obtain knowledge for some really difficult and incomprehensible topics. These simulations allow students to experience phenomena which could be dangerous, expensive or even impossible to observe in the real world.

In the old, jsXYZ application there were two offered experiments: Hook's law and Vibrating plane, in the section of Load experiments. These experiments were presented in the form of hyperlinks. In fact, that kind of presentation had many drawbacks, including that experiments were not sorted under some specific topics like: mechanics or electricity. At the same time, the user didn't know what he had chosen. When two students are working in pairs or when the teacher is presenting the simulation to the students, they won't know what exactly is presented in the cube. Another downside was the small number of simulations, which students can load directly. Figure 5.8, shows the look of the section Load experiments in the jsXYZ application.

Figure 5.8: Section of load experiments in jsXYZ application

In the redesigned application, xyzMobile the section of Load Experiments is presented entirely different. The renamed section, Show experiment is in the form of Pop-up widget. The content is opened in a popup. Because, the popup widget can be used for various types of popups, this popup is pre-

sented in the form of Nested menu, which can be created by placing listview into an collapsible inside the popup. In the Figure 5.9 it is showed how the button of show experiment, looks like.

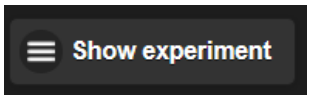

Figure 5.9: Button show experiment in xyzMobile application

When the pop-up widget is opened, 4 sections are displayed. The user can choose to perform simulation from one of the different classes of physical phenomenon: Electricity, Magnetism, Mechanics or Oscillation (Figure 5.10).

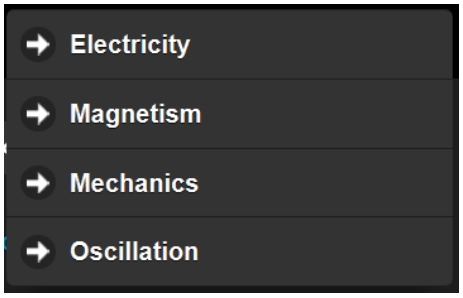

Figure 5.10: The pop-up widget of section Load experiments

Furthermore, when the user selects one of the classes of physical phenomenon additional menu is opened. This menu lists the simulations which are offered to the user. For example if the user selects the section of Mechanics, he has two experiments to choose from: free fall or pendulum.

Free fall simulation addresses the experiment of free falling objects which experience the same acceleration, independent of their mass. Also, to emphasize the fact that the inertial mass is changing proportional to the gravitational mass, the forces of inertia can be shown or hidden. The other

simulation, pendulum, helps to discuss the dynamics of velocity and acceleration during a full swing of the pendulum. Figure 5.11 presents the look of the nested menu with the experiments.

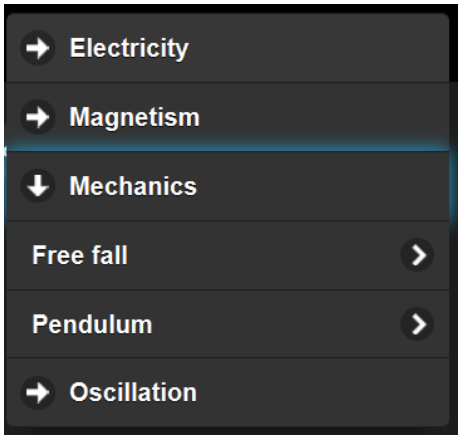

Figure 5.11: The nested menu with the experiments

Moreover, the application offers to the user three additional commands: run, stop and load file. The first two commands allow user to quickly repeat experiments as many times as he wants and rapidly explore the effect of many different parameters. The third of the commands is intended for reading simulations from the server. On the server user has a database of more than 30 physical experiments. Open file dialog is showed in Figure 5.12. Below, in Listings 5.2 and 5.3 is the source code of the dialog in HTML and its jQuery Mobile function.

```
<a href="#" class="upload ui-btn ui-mini ui-shadow-icon ui-btn-inline
 ui-icon-grid ui-btn-icon-left ui-corner-all" >Load file</a>
```

Listing 5.2: Code for open file dialog in HTML

```
$('.upload').on("click", function () {
  $('#readFile').click();
});
```

Listing 5.3: Function in jQuery Mobile for open file dialog

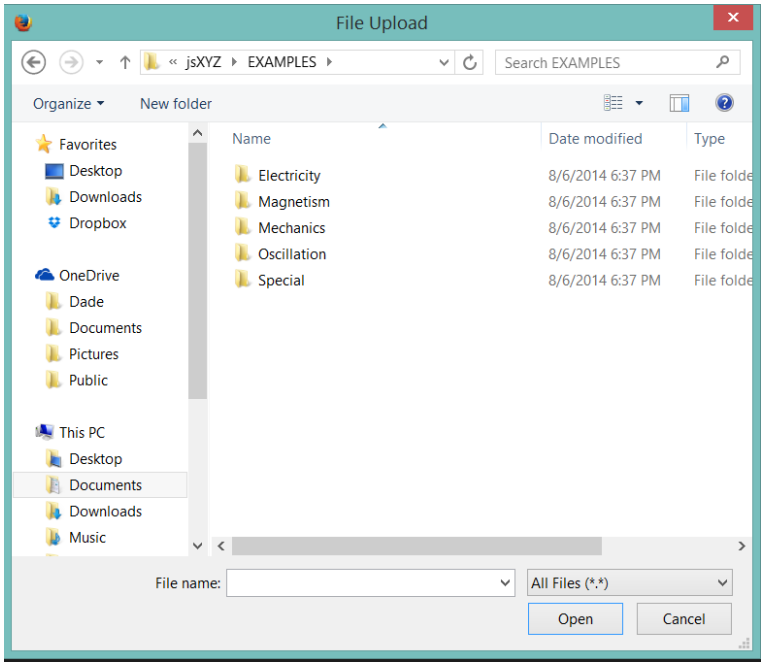

Figure 5.12: Open file dialog

Finally, we decided to implement a photo slider on a separate jQuery Mobile page. That slider is presenting six different pictures from simulations of the four sections: Electricity, Magnetism, Mechanics and Oscillation. These pictures show the user what he can expect from the simulations. The look of the photo slider is presented on the Figure 5.13. It is implemented in the code using the open-source Glide.js, available on its repository in GitHub [13]. Glide is responsive and touch-friendly jQuery slider that is simple, lightweight and fast.

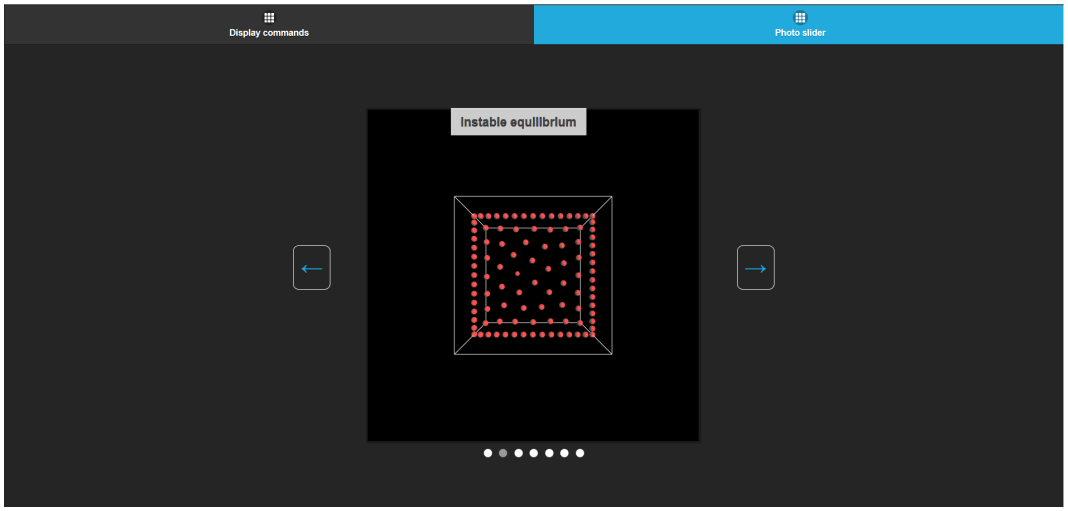

Figure 5.13: Photo slider

5.2 Support for multiuser physical simulations on mobile devices

Educational software development is no longer limited to developing single-user applications for deployment on desktop computers. Miniaturization and wireless communication advances have given us an ever- increasing array of portable devices, each with unique capabilities and form-factors. They range from small, lightweight devices, like cellular phones and handheld computers, to larger and more powerful machines, like laptops and tablet computers (Figure 5.14).

Even with the multitude of devices, most educational software still only targets one form-factor or another. There is no reason however, why applications cannot be built that span different devices, taking advantage of the different display size, portability, and computational power characteristics of the different devices [14]. Imagine that a group of students is trying to learn about the free fall. This simulation addresses the experiment of free falling objects which experience the same acceleration, independent of their mass. The students could gain additional understanding of the process by

simultaneously inspecting the objects with their handheld devices.

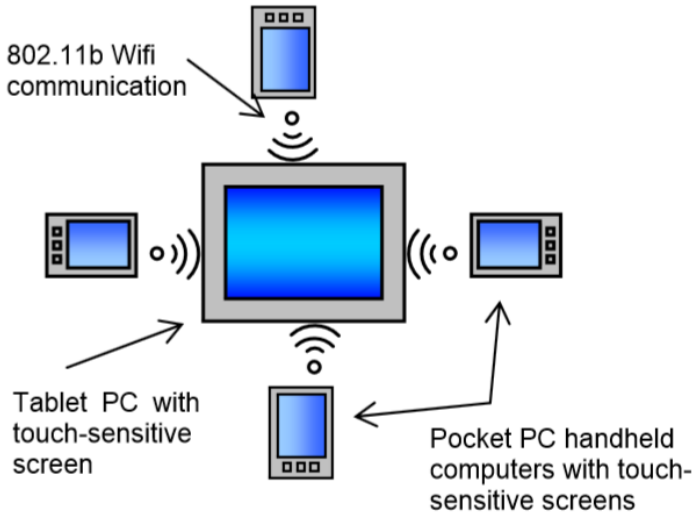

Figure 5.14: Topology of multiuser simulations

The xyzMobile application does just that, allowing students to independently observe and manipulate a shared real-time simulation. The students view and manipulate the simulation using their handhelds while simultaneously viewing and interacting with the entire simulation on a laptop or tablet style PC (Personal Computer). To achieve the multiuser real-time simulation by bi-directional communication between a browser and the server, we implemented WebSockets in our xyzMobile application.

5.2.1 WebSocket API

WebSockets provide a better interaction between a browser and xyzMobile application, facilitating live content and the creation of real-time simulations. This is made possible by providing a standardized way for the server to send content to the browser without being solicited by the client, and allowing for messages to be passed back and forth while keeping the connection open. In this way a two-way (bi-directional) ongoing conversation can take place

between a browser and the server.

Firstly, we started our implementation of WebSockets by detection of whether or not the client browser supports them. In case they are not supported, we have to choose another method of client-server communication. This step is realized by using the code in Listing 5.4.

```
if (!window.WebSocket) {
  alert("FATAL: WebSocket not natively supported. This will not work!");
}
```

Listing 5.4: Check if browser supports WebSockets

Furthermore, when WebSockets are supported by the browser, we needed to create a WebSocket object in order to communicate using the WebSocket protocol. In fact, this is done by connecting to a WebSocket server by calling the WebSocket constructor (Listing 5.5).

```
ws = new WebSocket("ws://" + hostName+ ":" + port);
```

Listing 5.5: Creating a WebSocket object

Optionally, instead of ws://, wss:// can be used. That is in fact secure socket variant to ws:// in the same way https is to http. Code re-written in that way is displayed in Listing 5.6.

```
ws = new WebSocket("wss://" + hostName+ ":" + port);
```

Listing 5.6: Creating a WebSocket object using wss://

Every WebSocet object can receive four different events that can be handled in the code:

- open – Event that is triggered on a client's side when socket connection is established.
- message – Event that is triggered when client receives data from server.

- close – Event that is triggered when connection is closed.
- error – Event that is triggered when there is an error in communication.

For this purpose we implemented four JavaScript event handlers: *onopen*, *onmessage*, *onclose* and *onerror*.

Onopen handler is needed on the client's side when the connection is accepted and established by the server. In xyzMobile application we are handling the open event with sending a message that a student has logged in (Listing 5.7).

```
ws.onopen = function() {
  ws.send("loginStudent");
}
```

Listing 5.7: Onopen event handler

If the connection is refused by the server or for some other reason the connection is closed, then the close event is fired. The *onclose* event handler is shown in Listing 5.8.

```
ws.onclose = function() {
  ws = null;
}
```

Listing 5.8: Onclose event handler

In case of any errors, they can be handled using the *onerror* event handler (Listing 5.9).

```
ws.onerror = function(error){
   ws.send('Error detected: ' + error);
}
```

Listing 5.9: Onerror event handler

5.3 JavaScript commands in xyzMobile application

There were many useful JavaScript functions initially written for the jsXYZ application that are also used in the new, redesigned xyzMobile application, such as setting the visibility of the cube or the coordinate system. The whole list of these functions and their brief description is presented in the Table 5.3. Another useful JavaSctipt functions allow us to control the particles. xyzMobile is a visually oriented tool that presents objects and structures in 2D or 3D space, which have the particles as their basic building elements. Particles are defined by their mass, charge, initial position and velocity and by using the JavaScript statements we can adjust these properties. The whole list of available statements is shown in the Tables 5.1 and 5.2.

Name	Meaning
mass	Mass of the particle
charge	Electric charge of the particle
restitution	% of retained bouncing velocity after collision
color	Color of the particle
radius	Size of the particle (its radius)
fixed	Boolean value if particle is fixed or not
traced	Boolean value, if particle is traced or not

Table 5.1: List of JavaScript statements for changing properties of particles

For example, the parameter of a particular (indexed) particle can be modified using the statement shown in Listing 5.10.

```
particles[i] . vx = 10;  // set x component oof velocity of particle i to 10
```

Listing 5.10: Modify the parameter of one particle

Name	Meaning
x	x position
y	y position
z	z position
vx	x component of velocity
vy	y component of velocity
vz	z component of velocity

Table 5.2: List of JavaScript statements for changing properties of particles

JavaScript commands	Brief description
addParticle(x, y, z, mass, charge, radius)	Add a new particle at specified position, mass, charge and radius
addSpring(i, j)	Add spring between 2 particles
checkParticlePosition(x, y, z)	Check the position x,yz for a existence of a particle
clearExperiment()	Clear experiment
exec(command_line)	Execute JavaXyzet command
freeze ()	Freeze all particles putting their velocities to 0
getNumParticles()	Returns the the number of currently created particles.
getNumSprings()	Returns the the number of currently created springs.
getSelectedParticleIndex()	Returns the eger index of the currently selected particle.
getSpringConst(i)	Get spring constant of spring with given index
getTime()	Returns current simulation time
getTimer()	Returns simulation timer (start stop timer)
memo()	Memorize current status of the simulation.
readFileFromServer(fileName)	Open experiment described in named file (relative to HTML page)
removeParticle (i)	Remove particle with index i and all springs connected to it
reset()	Restore the status of the simulation run to the previously memorized
selectFrontView ( )	Select front view to thescene
selecTopView ( )	Select top view to the scene
selectOrthographicCamera ( )	Select orthographic camera
selectPerspectiveCamera ( )	Select perspective camera
setAnimationRunning (boolean)	Start or stop the animation
setAVisible (boolean)	Set visibility of acceleration vectors
setAxesVisible(boolean)	Set visibility of the coordinate system
setCubeVisible(boolean)	Set visibility of XYZ cube
setEFieldParametersVisible(boolean)	Set visibility of electric field and potential parameters
SetFVisible(boolean)	Set visibility of F vectors
setFiVisible(boolean)	Set visibility of Fi vectors
setFloorVisible (boolean)	Set visibility of floor (as a thin panel)
setGlobalParametersVisible(boolean)	Set visibility of global parameters
setSpringConst(i, c)	Set spring constant to given value c
setSpringsEnabled (boolean)	Enable or disable all springs)
setSpringsVisible(boolean)	Set visibility ofall springs
setTimer(t)	Set simulation timer to value t
setTracelength(l)	Set length of traces for traced particles
setVVisible(boolean)	Set visibility of velocity vectors
startStopTimer()	Start / stop simulation timer (switch)

Table 5.3: List of JavaScript functions used in xyzMobile application

Chapter 6

Conclusion

In this thesis we created the application xyzMobile, which is an interactive, graphically oriented tool for visualizing physics simulations. It is supported by different devices, including mobile phones and tablets, taking advantage of the different display size, portability, and computational power characteristics of the devices.

Although the original program jsXYZ, on which xyzMobile was built, has been very useful for the preparation of interactive courses, there were some open issues that had to be addressed. Among them, probably the most important is that it could only be accessed via desktop computers and has not been designed using contemporary mobile-based technologies. In order to effectively redesign and alter the look of the application, we used jQuery Mobile technology. It is a framework that follows the open web standards and is compatible with a wide range of browsers and platforms. Considering the wide range of platforms that jQuery Mobile supported, we managed to make xyzMobile application reachable from as many devices as possible, including mobile devices and tablets. Furthermore, jQuery Mobile offered highly customizable UI, providing an optimal viewing experience that contributed the interface of our application to be easy for reading and navigation with a minimum of resizing, panning, and scrolling. We also provided the multiuser support of the simulations in real-time, offering users the possibility to si-

multaneously manipulate shared real-time simulations, by implementing the WebSockets. In fact, WebSockets offer bi-directional communication between a browser and the server. This is made possible by providing a standardized way for the server to send content to the browser without being solicited by the client, and allowing for messages to be passed back and forth while keeping the connection open.

We believe that modernizing the look and feel of the jsXYZ application, extending its usability on mobile and tablet devices and providing the support for multiuser simulations in real-time, has the potential to bring educational simulation tools to the whole new level.

Bibliography

[1] W. Winn and D Synder. Cognitive perspectives in psychology. *Handbook of research on educational communications and technology*, 1996.

[2] Coulomb's law. `http://en.wikipedia.org/wiki/Coulomb%27s_law`, August 2014.

[3] Javascript overview. `http://www.tutorialspoint.com/javascript/javascript_overview.htm`, April 2014.

[4] Brian Danchilla. *Beginning WebGL for HTML5.* August 2012.

[5] Tony Parisi. *Programming 3D Applications with HTML5 and WebGL.* February 2014.

[6] Three.js documentation. `http://threejs.org/docs/`, March 2014.

[7] Three.js wikipedia. `http://en.wikipedia.org/wiki/Three.js`, March 2014.

[8] Three.js github repository. `https://github.com/mrdoob/three.js/`, March 2014.

[9] jquery mobile api documentation. `http://api.jquerymobile.com/`, May 2014.

[10] Chetan K Jain. *jQuery Mobile Cookbook.* November 2012.

[11] Principle of user interface design. `http://en.wikipedia.org/wiki/Principles_of_user_interface_design#cite_note-1`, June 2014.

[12] Angela Chang, James Gouldstone, and Jamie Zigelbaum. Simplicity in interaction design. June 2014.

[13] Glide.js github repository. `https://github.com/jedrzejchalubek/Glide.js`, May 2014.

[14] Joseph Lee, Leilah Lyonss, Makiko Kawamura, and Chris Quintana. Mushi: Demonstrating a multi-user simulation with handheld integration. June 2006.

[15] Thomas M. Duffy and Donald J. Cunningham. 7. constructivism: Implications for the design and delivery of instruction. *Handbook of research on educational communications and technology*, 1996.

[16] P.Henning. Everyday cognition and situated learning. *Handbook of Research on Educational Communications and Technology*, 1998.

[17] Chen Der-Thanq and Hung W. L. David. Two kinds of scaffolding: The dialectical process within the authenticity-generalizibility (a-g) continuum. *Educational Technology and Society 5 (4)*, 2002.

[18] Les M. Lunce. Simulations: Bringing the benefits of situated learning to the traditional classroom. *Journal of Applied Educational Technology*, 3:37–45, 2006.

[19] H. Wenglinsky. Does it compute: The relationship between education technology and student achievement in mathematics, 1999.

[20] Stephen M.Alessi and Stanley R.Trollip. *Multimedia for learning: Methods and development (3rd Ed.)*. 2001.

[21] J. Peter Kincaid and Ken K. Westerlund. Simulaton in education and training. *Proceedings of the 2009 Winter Simulation Conference e*, 2009.